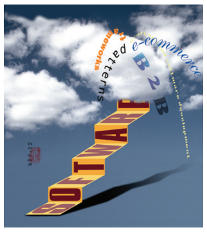
E-Business Process Modeling: The Next Big Step



The authors propose a Process Coordination Framework for Web services and outline the building blocks required for e-business automation. Their framework helps in understanding the roles of various standards and in identifying overlaps, gaps, and opportunities for convergence.

Selim Aissi
Pallavi Malu
Krishnamurthy Srinivasan
Intel Labs

The Internet is emerging as the platform for automating both the provider and consumer ends of e-business transactions. In this new model, businesses offer Web services that applications running in other businesses could invoke automatically, building bridges between systems that otherwise would require extensive integration and development efforts. Web services are software components that use standard Internet technologies to interact with one another dynamically.

In spite of the significant potential benefits, only a few large businesses have implemented automated e-business so far. Businesses realize that the cost of automating transactions with trading partners is very high. Standards and technologies for modeling e-business processes that use Web services could drive the costs down by achieving automated code generation, reuse, and interoperability—facilitating communication between business analysts and implementers.

Our proposed Process Coordination Framework outlines the building blocks required for Web services-enabled e-business automation. PCF helps in understanding the roles of the various proposed standards with respect to these building blocks and in identifying both overlaps and gaps.

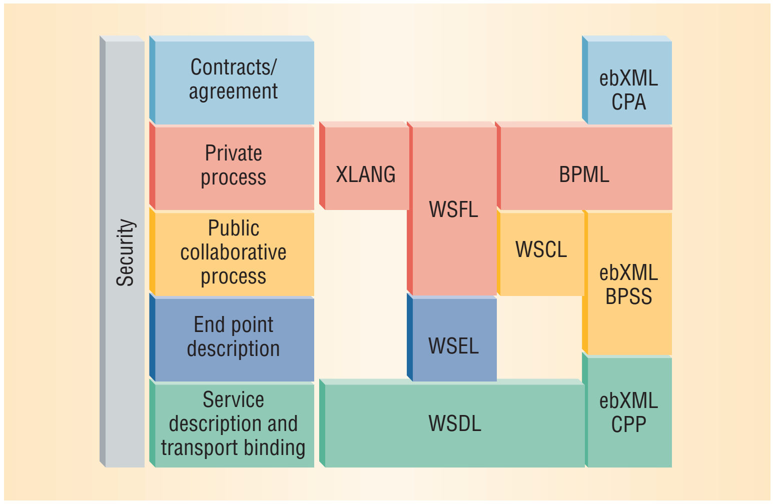


Figure 1. Process Coordination Framework. PCF groups features required for e-business automation along with relevant specifications into a multilayered stack.

PROCESS COORDINATION FRAMEWORK

As Figure 1 shows, PCF groups the features that e-business automation requires into a multilayered stack.

- *Service description and transport binding*. Service description provides metalevel data for services and their operations; transport bindings tie abstract service descriptions to specific physical addresses such as HTTP or SMTP statically at design time or dynamically at runtime.
- *End point description*. This layer describes aspects such as quality of service (QoS), service location, provider information, and service cost that can influence a customer's decision to use the Web service.
- *Public collaborative process*. This process describes the sequence or choreography of the

```
<message name="purchaseOrder"> <part name="body" type="xsd1:orderBody"/> </message>
<message name="shipmentNotification"> --------- </message>
<message name="payment"> --------- </message>
<message name="error"> --------- </message>
```
(a)

```
<portType name="TotalSupplyPortType">
   <operation name="processPO">
     <input message="tns:purchaseOrder"/>
     <output message="tns:shipmentNotification"/>
     <fault message="tns:error"/>
   </operation>
   <operation name="processPayment"> --------- </operation>
</portType>
```
(b)

```
<binding name="TotalSupplySoapBinding" type="tns:TotalSupplyPortType">
<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http/">
<operation name="processPO">
       <soap:operation soapAction="http://example.com/processPO"/>
        <input> <soap:body use="literal"/> </input>
        <output> --------- </output>
        <fault> <soap:fault use="literal"/> </fault>
</operation>
</binding>
```
(c)

```
<service name="TotalSupplyService">
   <documentation> TotalSupply service</documentation>
   <port name="TotalSupplyPort" binding="tns: TotalSupplySoapBinding">
     <soap:address location="http://example.com/TotalSupply"/>
   </port>
</service>
```
(d)

Figure 2. WSDL definition of TotalSupply: (a) schema file TotalSupply.xsd defining a class of XML documents; (b) portType element grouping the operation and related messages together; (c) binding element defining the network protocol details for the operation; (d) service element grouping related ports together.

operations the Web service supports. For example, if the service supports operations including Login and Check Balance, the public collaborative process specifies that the user needs to log in before checking the balance to avoid errors.

- *Private process*. This layer describes internal executable business processes that support the Web service's public collaborative process.
- *Contracts/agreement*. The actual technical and legal agreements under which entities conduct business reside in this layer. The technical contract between the entities includes the exact details of how the parties use their information technology (IT) infrastructure to transact the business at hand. The legal contract includes agreements on the terms and conditions of the business exchange. Having an XML-based syntax to specify contracts helps applications automatically interact without human intervention. However, XML does not currently incorporate many of these definitions.
- *Security*. Security requirements include a combination of the following features: authorization, authentication, confidentiality, nonrepudiation, and auditing. Various PCF layers may require different aspects of security. Any exchange of business information may require all, some, or no security features.

SPECIFICATIONS IN THE PCF SPACE

A simple example of a composite Web service provided in IBM's Web Services Flow Language (http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf) helps to understand the specifications and the features they support. In this example, TotalSupply service leverages two other Web services: mySupplier as a source for products and myShipper to distribute them. These two Web services use the Web Services Description Language.

Web Services Description Language

WSDL is an XML-based specification that describes Web services as collections of message-enabled operations and was submitted to the World

```
<ConversationInteractions>
   <Interaction StepType="ReceiveSend" id="ProcessPO">
     <InboundXMLDocuments>
       <InboundXMLDocument id="PurchaseOrder"
             hrefSchema="PO.xsd"/>
     </InboundXMLDocuments>
   <OutboundXMLDocuments>
     <OutboundXMLDocument id="Shipment"
             hrefSchema="Shipment.xsd"/>
     </OutboundXMLDocuments>
   </Interaction>
   <Interaction StepType="Receive" id="ProcessPayment"> --------- </Interaction>
   </ConversationInteractions>
<ConversationTransitions>
  <Transition>
   <SourceInteration href="#ProcessPO"/>
   <DestinationInteraction href="#ProcessPayment"/>
  <TriggeringDocument href="#Shipment"/>
</Transition>
</ConversationTransitions>
```

Figure 3. Web Services Conversation Language defining conversations between service providers and consumers. The Transition attribute defines the ordering between the processPO and processPayment interactions.

Wide Web consortium (W3C) by Ariba, IBM, and Microsoft (http://www.w3.org/TR/wsdl). WSDL provides the information needed to invoke an operation in an application running in a third-party location on the Internet, including the operation name, input/output parameters and their data types, the network protocol, and the network address.

Dissecting the definition of TotalSupply helps to understand the details of WSDL. The service supports two operations: processPO and processPayment. This example assumes that the service is deployed using the SOAP protocol (http://www.w3.org/TR/SOAP/) over HTTP. The WSDL file begins with an element having attributes that define the TotalSupply service, the target namespace (http://example.com/TotalSupply.wsdl), and other namespaces. The types element defines the data type of the input/output messages that are exchanged during the operation. You can use standard XML data types such as string, integer, and float, or you can create complex data types.

TotalSupply supports four data types: orderBody, notificationBody, paymentBody, and errorBody. It also supports four types of single-part messages: two input messages, purchaseOrder and payment; one output message, shipmentNotification; and an error message. To keep the example simple, we define a class of XML documents within a separate schema file, TotalSupply.xsd, as Figure 2a shows. The references to the user-defined types use xsd1 as a namespace prefix to qualify their use. The message element defines the exchanged data. Messages can consist of one or more logical parts, each with an associated data type.

The portType element in Figure 2b groups the operation and related messages together. This element currently supports four types of operation: one-way, request-response, solicit response, and notification. TotalSupply Web service supports two request-response operations: processPO and processPayment. The binding element in Figure 2c defines the network protocol details for the operation, SOAP over an HTTP transport in this example. Where the port element defines a single network address (http://example.com/TotalSupply), the service element in Figure 2d groups related ports together in our example Web service.

Web Services Conversation Language

Hewlett-Packard developed the Web Services Conversation Language (WSCL), an XML-based specification layered on top of WSDL, for use in defining conversations between service providers and consumers. WSCL was submitted to the W3C (http://www.w3.org/TR/wscl10/).

Figure 3 shows a WSCL conversation for our TotalSupply Web service. The initialInteraction attribute specifies the first interaction—equivalent to operation in WSDL—in the sequence. The Transition attribute defines the ordering between the processPO and processPayment interactions. The conversation proceeds from one interaction to another according to the legally defined transitions. Transition is unique to WSCL, whereas Interaction and Inbound/Outbound XML documents overlap with WSDL.

ebXML

The United Nations (UN/CEFACT) and OASIS sponsored the ebXML specifications for use in e-business frameworks.

ebXML Business Process Specification Schema. ebXML BPSS (http://www.ebxml.org/specs/ebBPSS.doc) is a proposed standard for specifying collaborations for use in exchanging business documents

Figure 4. ebXML Business Process Specification Schema specifying business collaborations. In business collaborations, a set of roles exchange business documents in a set of choreographed transactions.

```
   <!--Business Transactions-->
   <BusinessTransaction name="Process PO">
   <RequestingBusinessActivity name="PurchaseActivity">
<DocumentEnvelope businessDocument="Purchase Order"
isAuthenticated="true"/>
   </RequestingBusinessActivity>
   <RespondingBusinessActivity name="NotificationActivity">
<DocumentEnvelope businessDocument="Shipment Notification"
isTamperProof="true"/>
   </RespondingBusinessActivity>
   </BusinessTransaction>
   <BusinessTransaction name="Process Payment">
   <RequestingBusinessActivity name="PaymentActivity">
<DocumentEnvelope businessDocument="Payment"
isAuthenticated="true"/>
   </RequestingBusinessActivity>
   </BusinessTransaction>
```

through a set of choreographed transactions. Our TotalSupplySpec example uses three business documents: purchase order, shipment notification, and payment.

A business transaction is an atomic unit of work between trading partners. Each business transaction has one requesting (incoming) document and an optional responding (outgoing) document. BPSS also supports business signals, or application-level documents that signal a business transaction's current state—for example, an acknowledgment document.

As Figure 4 shows, TotalSupplySpec supports two business transactions: processPO and processPayment. TotalSupply workflow supports the notion of business collaboration, or BinaryCollaboration, which establishes roles and authorizes actors to participate in the collaboration. For example, a Buyer role can initiate the business transaction and the Seller has a responding role. A BinaryCollaboration also specifies the choreography—or ordering of business transactions—and subcollaborations within larger collaborations. WSCL overlaps with BPSS and WSFL because Transition is the same as choreography.

BPSS uses a top-down approach that provides a notation to describe a business process from both the service provider and service consumer viewpoints, whereas WSDL and WSCL use a bottom-up approach that provides a notation describing only the service provider interface.

ebXML Collaboration Protocol Profile. ebXML CPP (http://www.ebxml.org/specs/ebCCP.doc) describes the IT capabilities of an individual party participating in the collaborative business process. These capabilities include details of transport, messaging, security constraints, and bindings to a business process—for example, ebXML BPSS.

A CollaborationProtocolProfile, the root element of a CPP document, includes a PartyInfo element that identifies the party described in the CPP and includes a reference to the ProcessSpecification (BPSS) documents it supports. The CollaborationRole element identifies the roles that the party can play in the referenced business process. ServiceBinding binds all of the BusinessTransactions defined in BPSS to the sending or receiving communication protocol—for example, HTTP or SMTP—and also provides the end point associated with the receiving protocol.

In the current CPP specification, the end point attribute is static, but the ebXML Messaging Service specification may provide the capability to dynamically override that end point information by exchanging a specific URI in a business document.

The Collaboration Protocol Agreement (CPA) describes the capabilities that two parties have agreed to use to perform a business collaboration. A CPA can be generated by calculating the intersection of the information found in the CPPs of the two parties participating in the collaboration. Generating the CPA may involve some level of negotiation between the two parties.

CPA is part of the CPP specifications. Although there is no specification for how to accomplish the negotiation, the CPPA technical committee has initiated some work in this area.

Web Services Flow Language

IBM's Web Services Flow Language is an XML-based specification for describing a public collaborative process and its compositions. WSFL is layered on top of WSDL, which describes the service interfaces and their protocol bindings.

WSFL defines two types of Web services compositions: A flow model specifies the execution sequence of a business process's functions. A global model combines flow models and provides a description of how the composed Web services interact with each other.

WSFL's major features include

- *activity*, a processing step in a business process;
- *control link*, sequencing rules in a business process that model the control flow from one activity to the next;
- *data link*, information flow in a business process, wherein data flow can be separate from control flow;
- *data mapping*, specifying information that needs to be transferred between two linked activities; and
- *pluglink* and *export element*, describing the relation between activities in the flow model and the WSDL operations the service provider offers.

Figure 5 shows a graphical representation of the TotalSupply Web service WSFL model, with three flow models comprising the global model: one for our enterprise and two representing the supporting services.

Prior to constructing a WSFL global model, the individual flow models must be specified according to IBM's WSFL specification (http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf). The WSFL flow model for TotalSupply consists of a flowModel name, TotalSupplyFlow; a service ProviderType; and a listing of service providers, activity specifications, and control and data links. Our TotalSupply flow model includes two separate serviceProvider elements in the service ProviderType—mySupplier and myShipper. Figure 6a shows the XML code for mySupplier.

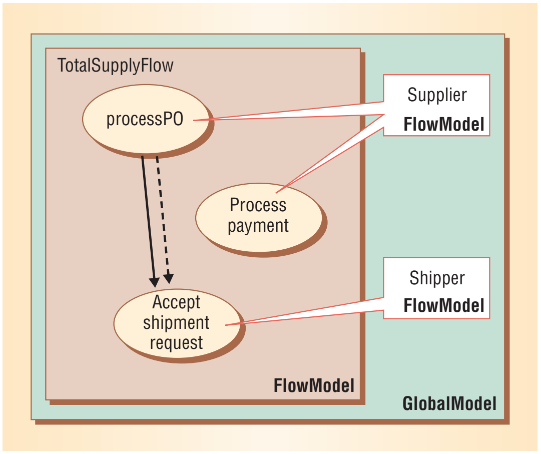


Figure 5. TotalSupply Web service global model representation. Activities such as processPO are represented as shaded circles, control links are solid arrows connecting activities, data links are dashed arrows, and the callouts connecting the TotalSupply activities to the external services represent pluglinks.

The TotalSupply flow model must perform three activities to successfully complete the business process: processing a purchase order, accepting a shipment request, and receiving a payment. Separate activity elements specify each of these activities, which must perform in a specific order: Purchase-order processing must precede the shipper's acceptance of the shipping request, whereas the payment can be received at any time—WSFL allows an activity to exist outside the control links connecting other activities. Figure 6b provides an example of the processPO activity, which is provided by my-

```
<serviceProvider name="mySupplier" type="supplier">
  <locator type="static" service="qualitySupply.com"/>
</serviceProvider>
```
(a)

```
<activity name="processPO">
  <performedBy serviceProvider="mySupplier"/>
  <implement>
    <export>
      <target portType="totalSupplyPT"
          operation="sendProcOrder"/>
    </export>
  </implement>
</activity>
```
(b)

```
<controlLink source="processPO"
      target="acceptShipmentRequest"/>

<dataLink source="processPO"
        target="acceptShipmentRequest">
  <map sourceMessage="anINVandSR" targetMessage="anSR"/>
</dataLink>
```
(c)

Figure 6. WSFL TotalSupply flow model: (a) serviceProvider element for mySupplier; (b) processPO activity provided by mySupplier; (c) controlLink and dataLink elements.

Figure 7. Simplified global model mySupplyChain captures the interactions between the supplier, the shipper, and the TotalSupply Web service.

```
<globalModel name="mySupplyChain"
       serviceProviderType="supplyChain">
   <serviceProvider name="mySupplier" type="supplier"/>
   <serviceProvider name="myShipper" type="shipper"/>
   <serviceProvider name="myTotalSupply" type="totalSupply"/>
   <plugLink>
     <source serviceProvider="myTotalSupply"
       portType="totalSupplyPT"
       operation="sendProcOrder"/>
     <target serviceProvider="mySupplier"
       portType="suppSvr"
       operation="procPO"/>
   </plugLink>
   <plugLink>
     <source serviceProvider="myTotalSupply"
       portType="totalSupplyPT"
       operation="sendPayment"/>
     <target serviceProvider="mySupplier"
       portType="suppSvr"
       operation="recPay"/>
   </plugLink>
   <plugLink>
     <source serviceProvider="myTotalSupply"
       portType="totalSupplyPT"
       operation="sendSR"/>
     <target serviceProvider="myShipper"
       portType="shipSvr"
       operation="recSR"/>
   </plugLink>
</globalModel>
```

Supplier, and exposes the sendProcOrder operation; the TotalSupply flow includes this activity definition as well as definitions for acceptShipmentRequest and processPayment.

Figure 6c provides an example of the controlLink and dataLink elements. The map element nested inside the dataLink specifies the information the two linked activities need to transfer.

All of this is brought together in Figure 7 in which the simplified mySupplyChain global model captures the interactions between the supplier, the shipper, and the TotalSupply Web service. The relationship between a flow model public interface operation and a service provider operation is established through a plugLink element, which WSFL typically specifies within a global model.

XLANG

Microsoft's XLANG (http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm), an XML-based specification for describing executable business processes internal to a business, is layered on top of WSDL.

The XLANG specification builds on the XML code for process description that Microsoft's Visio-based BizTalk Server Orchestration graphical modeling tool generates.

XLANG's major features include

- *behavior*, a container for the description of the service's behavioral aspects, including support for looping, concurrency, and exception handling;
- *action*, atoms of behavior referencing WSDL operations on the available ports;
- *control flow*, sequence in which the service performs actions;
- *correlation*, structure the service uses to route messages to correct workflow instances;
- *context*, a context for long-running transactions;
- *service management*, features of service instance management; and
- *port mapping*, method for plugging in the service user and the service provider.

Figure 8a shows an XLANG definition for our WSFL-based TotalSupply example. The </all> tag indicates that acceptShipmentRequest and processPayment actions can occur concurrently. WSDL defines the sendSRPort and sendPaymentPort reference ports. Figure 8b captures the relationship between the supplier, the shipper, and the TotalSupply Web service. A connect element establishes

```
<xlang:behavior>
  <xlang:body>
    <xlang:sequence>
      <xlang:action operation="processPO"
             port="sendProcOrderPort" activation="true"/>
             <all>
      <xlang:action operation="acceptShipmentRequest"
                        port="sendSRPort"/>
      <xlang:action operation="processPayment"
                        port="sendPaymentPort"/>
             </all>
      </xlang:action>
    </xlang:sequence>
  </xlang:body>
</xlang:behavior>
```
(a)

```
<!--use the WSDL import for convenience-->
<import namespace="http://example.com/totalSupply/definitionsSupplier"
location="http://example.com/totalSupply/definitionsSupplier.wsdl"/>
<import namespace="http://example.com/totalSupply/definitionsShipper"
location="http://example.com/totalSupply/definitionsShipper.wsdl"/>
<import namespace="http://example.com/totalSupply/definitionsTotalSupply"
location="http://example.com/totalSupply/definitionsTotalSupply.wsdl"/>
<!--define the way in which the services plugged in together-->
 <xlang:contract>
<xlang:services refs="totalSupply:totalSupplyService
supplier:SupplierService
shipper:ShipperService"/>
 <xlang:portMap>
<!--the total supply connects to the supplier for PO and Payment-->
<xlang:connect
port="totalSupply:totalSupplyService/sendProcOrderPort"
port="supplier:SupplierService/suppSvr/procPOPort"/>
<xlang:connect
port="totalSupply:totalSupplyService/sendPaymentPort"
port="supplier:SupplierService/suppSvr/recPayPort"/>
<!--the total supply connects to the shipper for the shipment request -->
<xlang:connect
port="totalSupply:totalSupplyService/sendSRPort"
port="shipper:ShipperService/shipSvr/recSRPort"/>
</xlang:portMap>
</xlang:contract>
```
(b)

Figure 8. (a) XLANG definition for TotalSupply; (b) relationship between the supplier, the shipper, and the TotalSupply Web service, established through a connect element.

the relationship between a TotalSupply operation and an operation that a Supplier and a Shipper provide.

Business Process Modeling Language

The Business Process Management Initiative developed the XML-based BPML metalanguage for modeling executable private business processes (http://www.bpmi.org/bpml.esp). BPML is complementary to public collaborative process description languages, such as BPSS. It is based on the concept of transactional finite-state machines and has features that overlap XLANG.

CHALLENGES

The abstract public description of our TotalSupply Web service example needs to include a definition of some additional attributes, including constraints the B2B environment's competitive nature imposes, which requires differentiations based on QoS.

End point description specification. We need to standardize and implement these attributes to allow an optional layering of the definitions in WSDL. This would allow an implementer to add the complexity of this layer only when the need arises. Currently, there is no standard form for these definitions. Although IBM's WSFL documentation hints at work in this area, for example, WSEL, no standards body has pursued these efforts.

Security considerations. The biggest challenge for Web services is the fragmentation of the security requirements. The inability to describe where and how to apply security measures is a large gap in the description of Web services. The only initiative that indicates how to use XML's security features in a CPPA context is ebXML. The main security challenges include the lack of specificity for how to apply security standards (for example, digital certificates); the cost or difficulty in implementing some security solutions (for example, a digital signature for authentication); and the lack of key security countermeasures in the specification (for example, security policy maintenance).

Another challenge is that ebXML CPPA's security features are intermingled with other trading partner specific details like business process definitions and implementation protocol bindings. This approach works well when every layer of the ebXML protocol stack is used. However, with ebXML, the implementer cannot use widely supported standards from other organizations.

Opportunities for convergence. Having multiple specifications that overlap and describe similar features can cause interoperability problems. Instead of describing its own transport bindings, CPPA could use WSDL's transport bindings.

Both XLANG and BPML focus on an XML-based description of private business processes. Because they both are built on WSDL and have relative gaps and strengths, convergence would capitalize on their strengths. WSCL describes the sequence in which operations can be invoked.

WSFL specifies public choreography and combining multiple Web services into a composite Web service. BPSS describes public choreography and multiparty collaboration. WSCL, WSFL, and BPSS all describe the public choreography of a business process, and they should converge on that feature.

Based on its B2B focus, security and end point descriptions, and complete business-scenario approach, ebXML (BPSS, CPPA) tends to be the richest of all the specifications discussed. However, to reach full automation, in addition to security descriptions, Web services need a richer and more complete end point. Web services also need faster, less expensive, and more modular business process modeling and code generation tools, such as graphical tools.

In a perfect world, the analyst draws the model, the tools generate the WSDL and WSCL or BPSS, and the implementer uses the graphical model to derive the workflow implementation and automatically generate the CPP, which the trading partners then use to do business. While XML allows interoperability, the proposed automation process will facilitate reuse of the workflows.

Web services are advancing platform and language-independent interoperability. However, their standardized definitions are still evolving. Describing the Web services business process is a key area in the future of software engineering. In addition to discovery and management, a Web services description will lead to solutions that are more flexible, faster to implement, and cheaper to deploy and maintain. ■

Selim Aissi *is a senior architect in Intel's Corporate Technology Group in Hillsboro, Oregon. His research interests include the development of security reliability, and trustworthiness technologies for distributed systems and Web services. He received a PhD in aerospace engineering from the University of Michigan. Contact him at selim.aissi@intel.com.*

Pallavi Malu *is a senior software engineer in Intel's Corporate Technology Group in Chandler, Arizona. She received an MS in computer science from Wright State University. Contact her at pallavi.g.malu@intel.com.*

Krishnamurthy Srinivasan *is a manager in the Web Services Technologies Lab at Intel's Corporate Technology Group. He received a PhD in textile engineering from the Georgia Institute of Technology. Contact him at krishnamurthy.srinivasan@intel.com.*